TS

OpenWeather

Equipo 09

Open Weather API

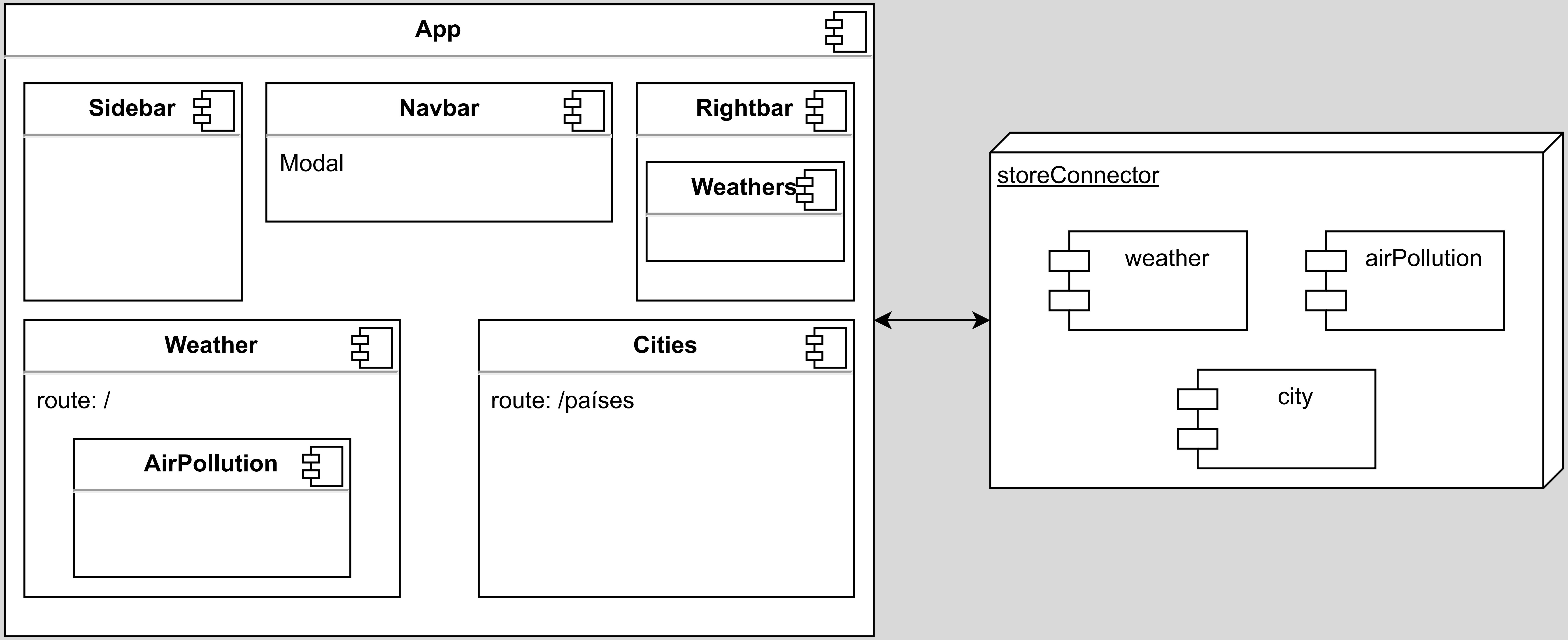
Se utilizaron consultas de:

- Temperaturas: próximos 5 días cada 3 horas
- Contaminación del aire: actual y próximos 5 días cada hora

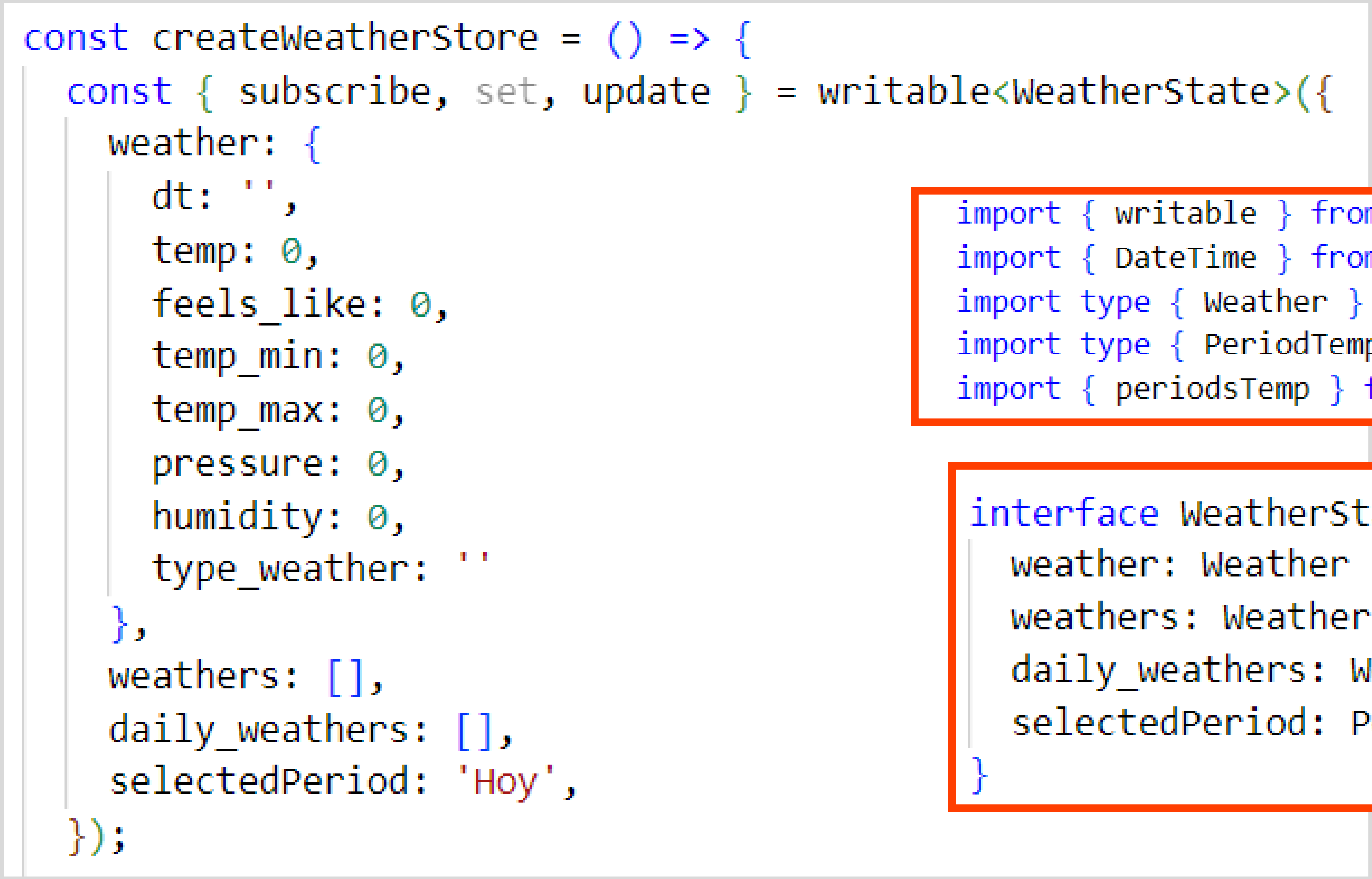
OpenWeather

Demo

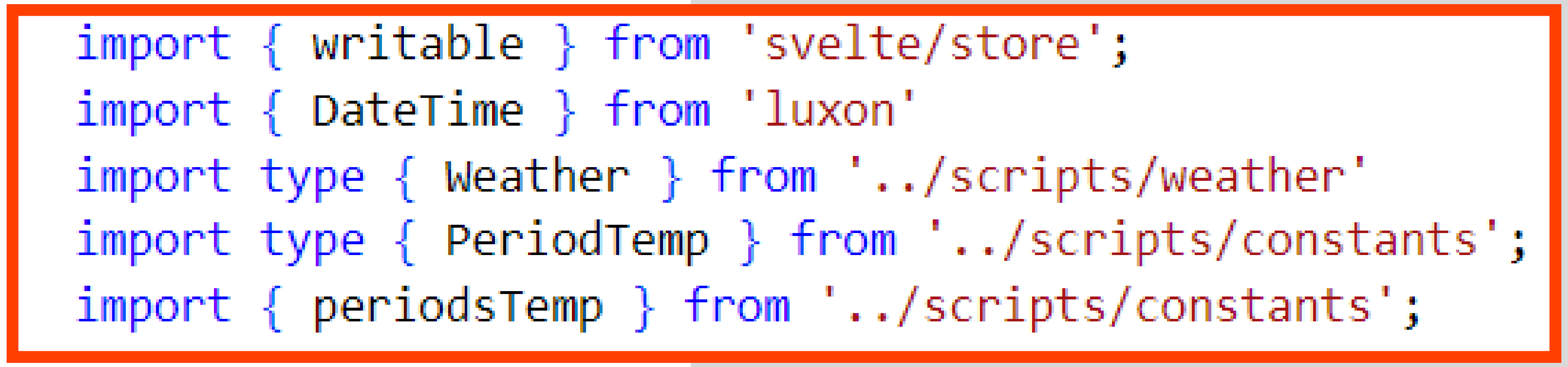
Arquitectura



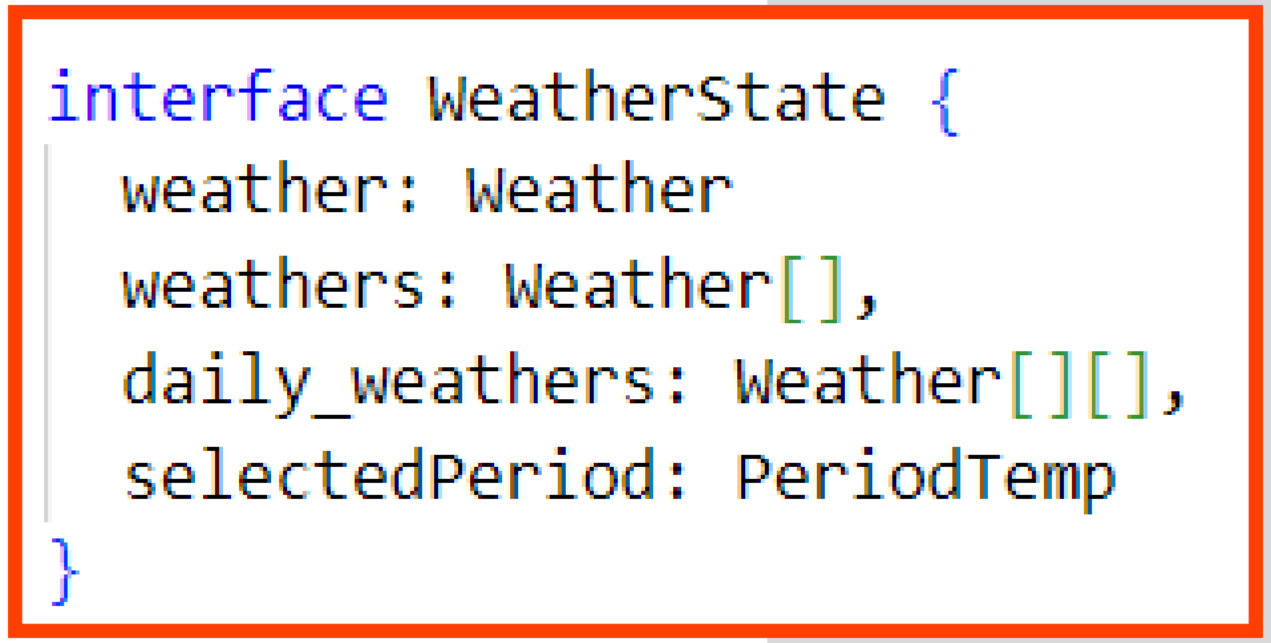
Store: weather.ts



```ts
const createWeatherStore = () => {
  const { subscribe, set, update } = writable<WeatherState>({
    weather: {
      dt: '',
      temp: 0,
      feels_like: 0,
      temp_min: 0,
      temp_max: 0,
      pressure: 0,
      humidity: 0,
      type_weather: ''
    },
    weathers: [],
    daily_weathers: [],
    selectedPeriod: 'Hoy',
  });
```

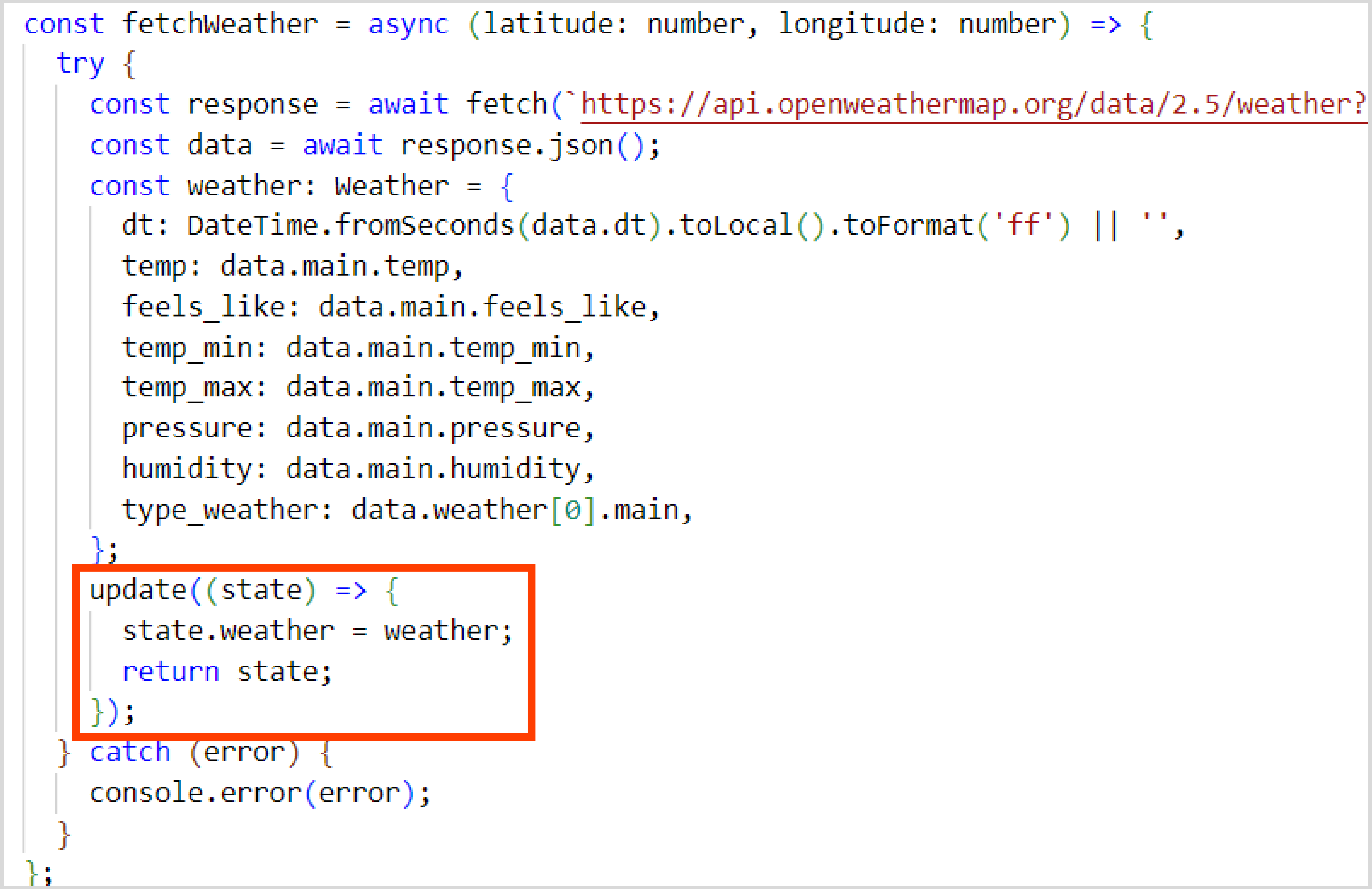


```ts
import { writable } from 'svelte/store';
import { DateTime } from 'luxon'
import type { Weather } from '../scripts/weather'
import type { PeriodTemp } from '../scripts/constants';
import { periodsTemp } from '../scripts/constants';
```



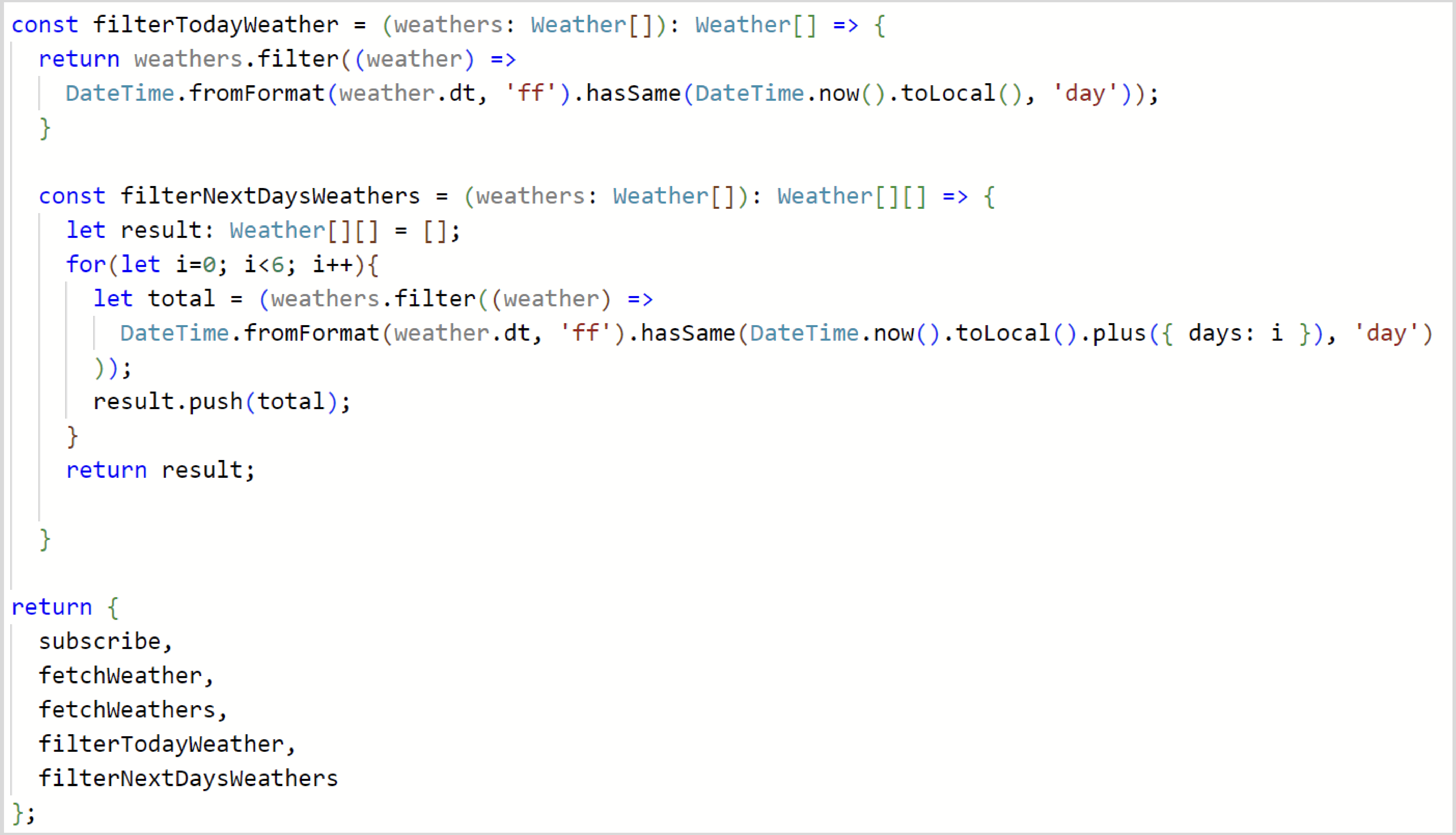
```ts
interface WeatherState {
  weather: Weather
  weathers: Weather[],
  daily_weathers: Weather[][],
  selectedPeriod: PeriodTemp
}
```

Store: weather.ts



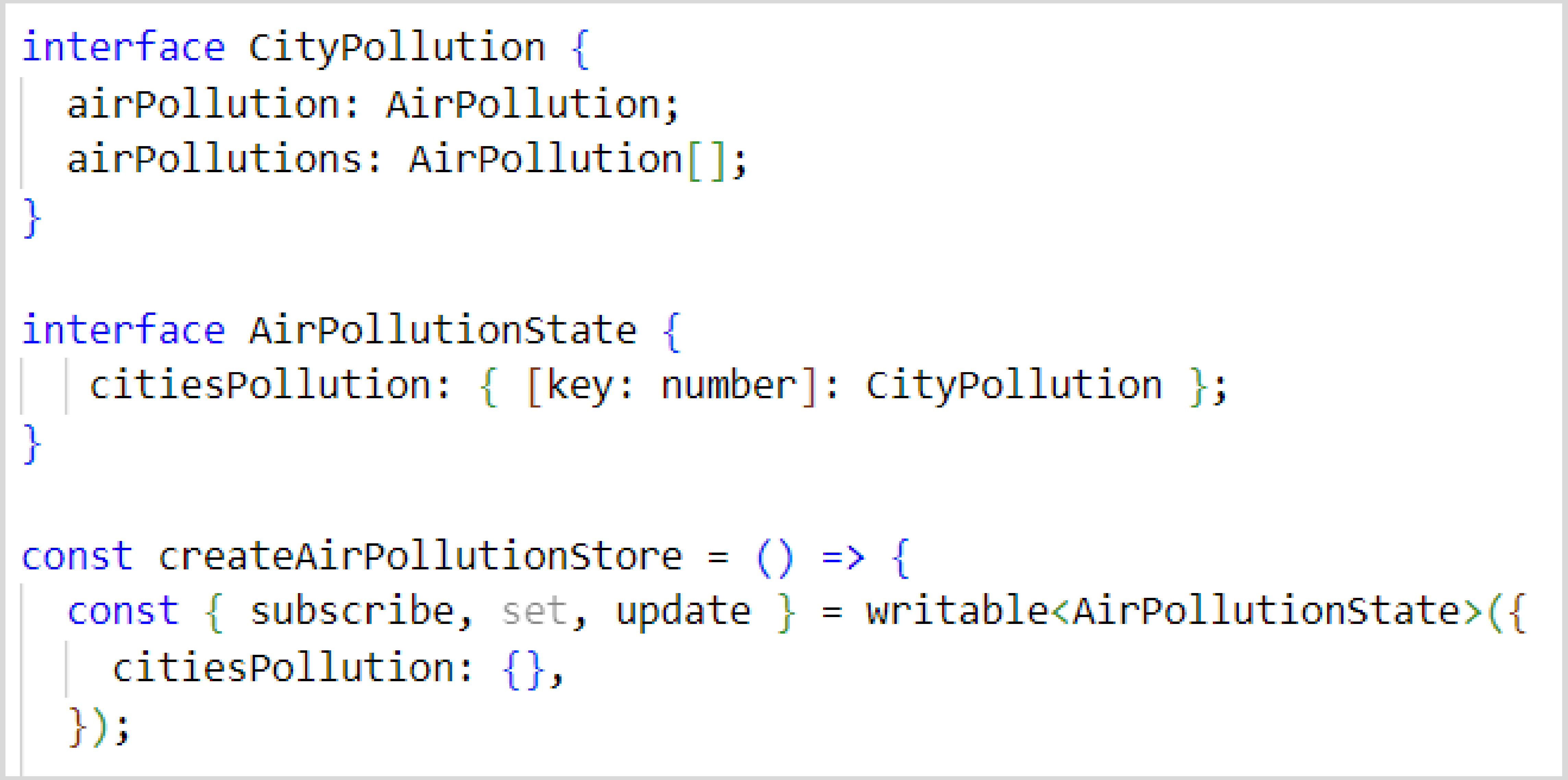
```
const fetchWeather = async (latitude: number, longitude: number) => {
  try {
    const response = await fetch(`https://api.openweathermap.org/data/2.5/weather?
    const data = await response.json();
    const weather: Weather = {
      dt: DateTime.fromSeconds(data.dt).toLocal().toFormat('ff') || '',
      temp: data.main.temp,
      feels_like: data.main.feels_like,
      temp_min: data.main.temp_min,
      temp_max: data.main.temp_max,
      pressure: data.main.pressure,
      humidity: data.main.humidity,
      type_weather: data.weather[0].main,
    };
    update((state) => {
      state.weather = weather;
      return state;
    });
  } catch (error) {
    console.error(error);
  }
};
```

Store: weather.ts



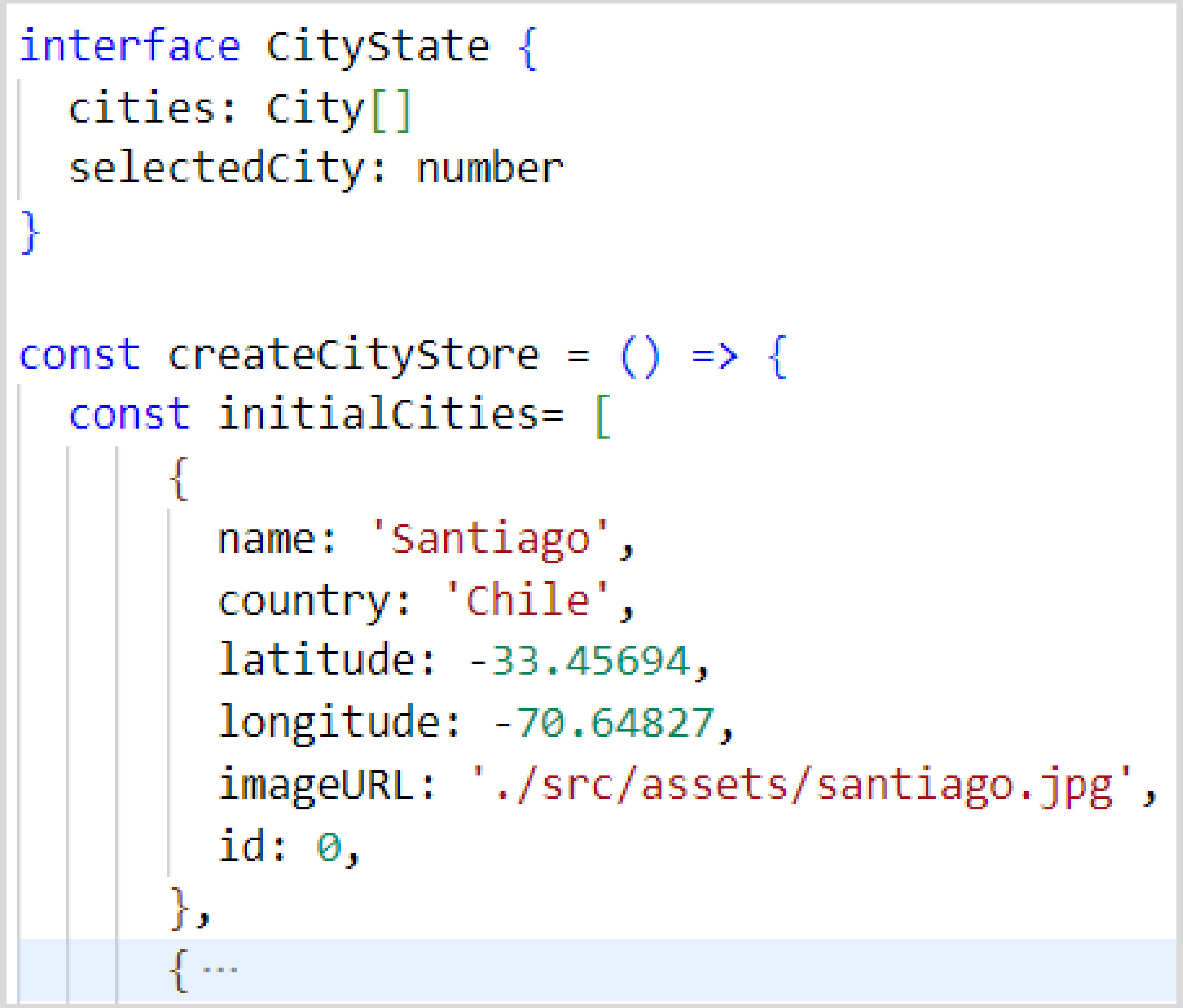
```
const filterTodayWeather = (weathers: Weather[]): Weather[] => {
  return weathers.filter((weather) =>
    DateTime.fromFormat(weather.dt, 'ff').hasSame(DateTime.now().toLocal(), 'day'));
  }

  const filterNextDaysWeathers = (weathers: Weather[]): Weather[][] => {
    let result: Weather[][] = [];
    for(let i=0; i<6; i++){
      let total = (weathers.filter((weather) =>
        DateTime.fromFormat(weather.dt, 'ff').hasSame(DateTime.now().toLocal().plus({ days: i }), 'day')
      ));
      result.push(total);
    }
    return result;

  }

return {
  subscribe,
  fetchWeather,
  fetchWeathers,
  filterTodayWeather,
  filterNextDaysWeathers
};
```

Store: airPollution.ts

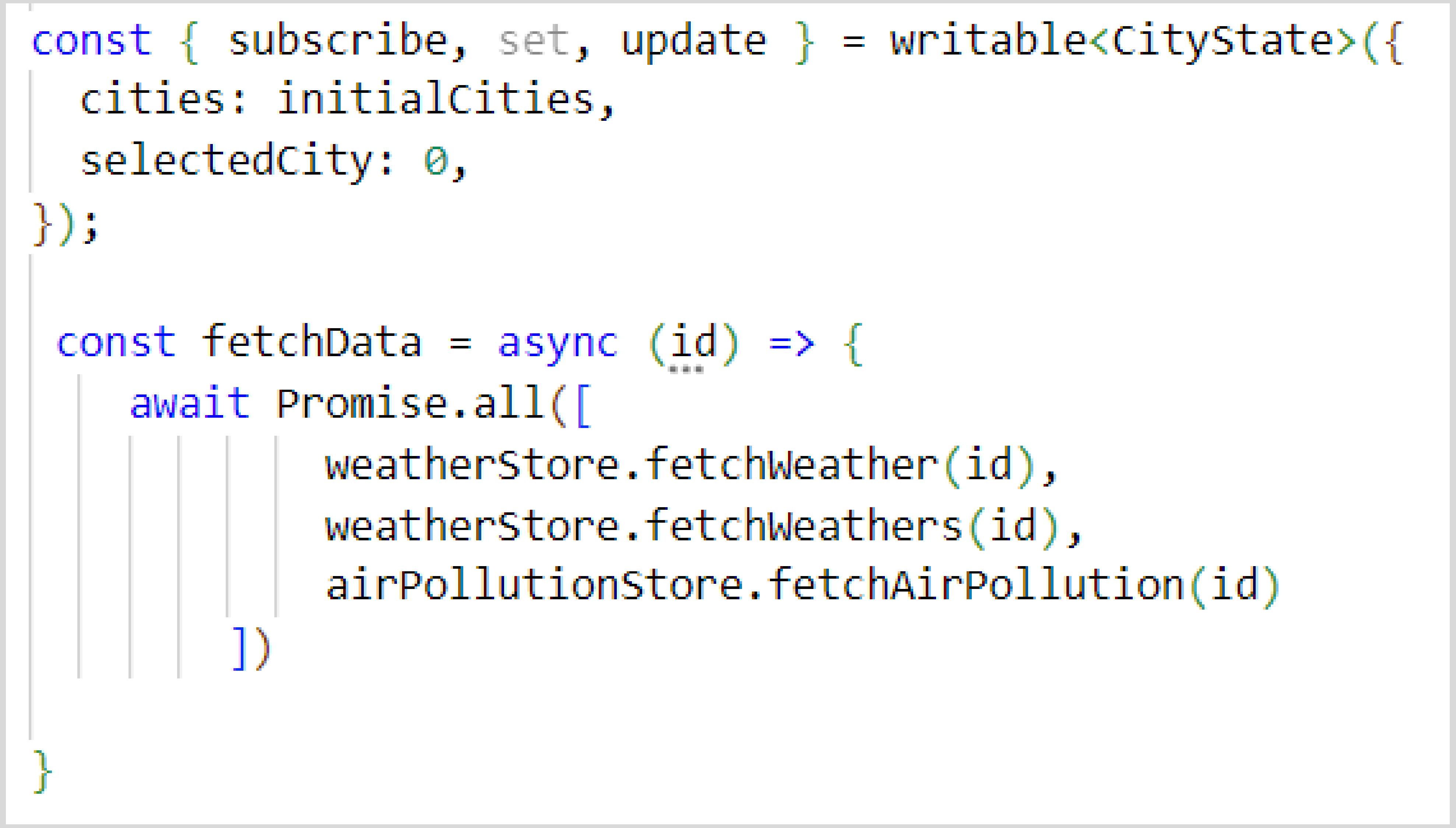


```typescript
interface CityPollution {
  airPollution: AirPollution;
  airPollutions: AirPollution[];
}

interface AirPollutionState {
   citiesPollution: { [key: number]: CityPollution };
}

const createAirPollutionStore = () => {
  const { subscribe, set, update } = writable<AirPollutionState>({
    citiesPollution: {},
  });
```

Store: city.ts

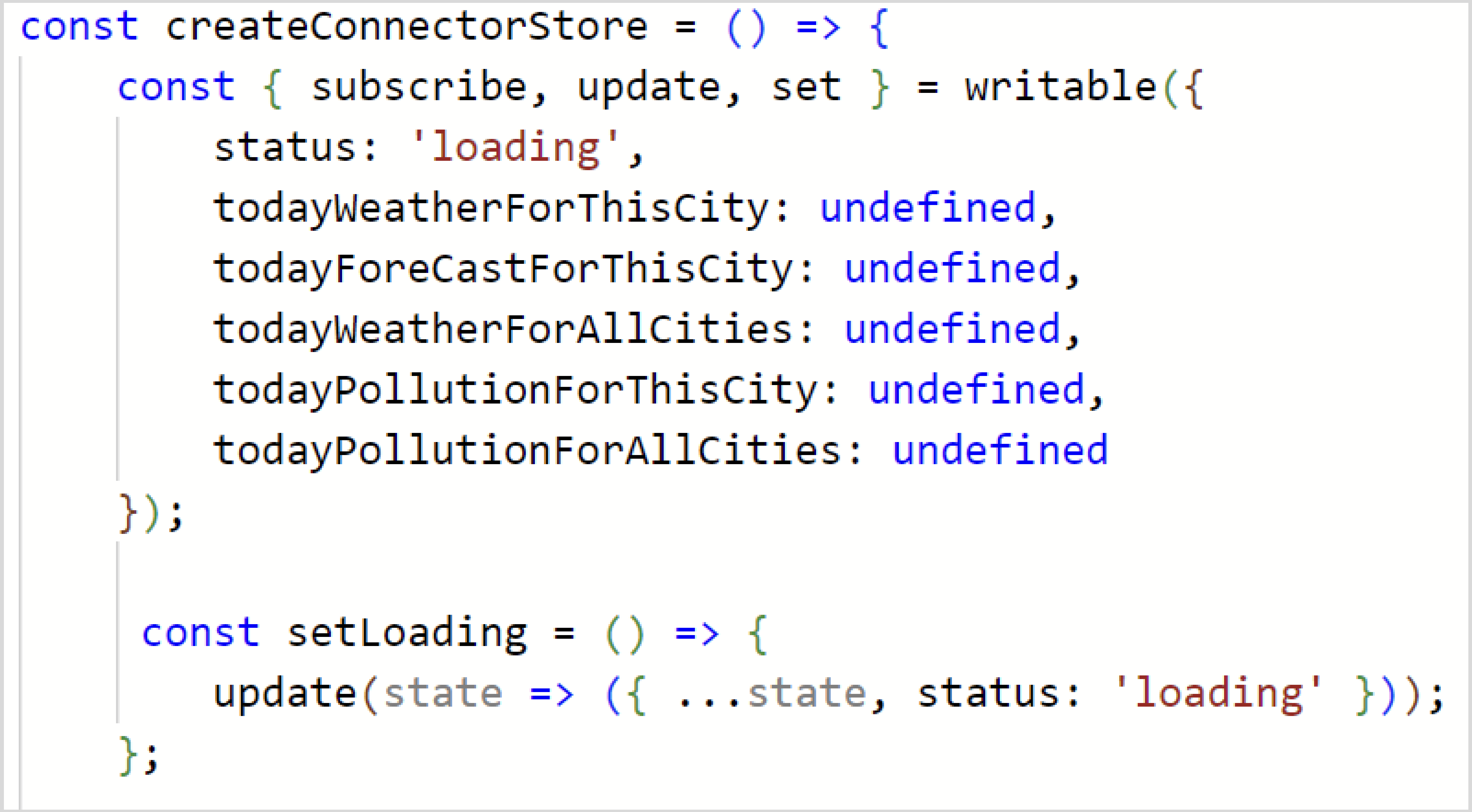


```
interface CityState {
  cities: City[]
  selectedCity: number
}

const createCityStore = () => {
  const initialCities= [
      {
        name: 'Santiago',
        country: 'Chile',
        latitude: -33.45694,
        longitude: -70.64827,
        imageURL: './src/assets/santiago.jpg',
        id: 0,
      },
      {…
```



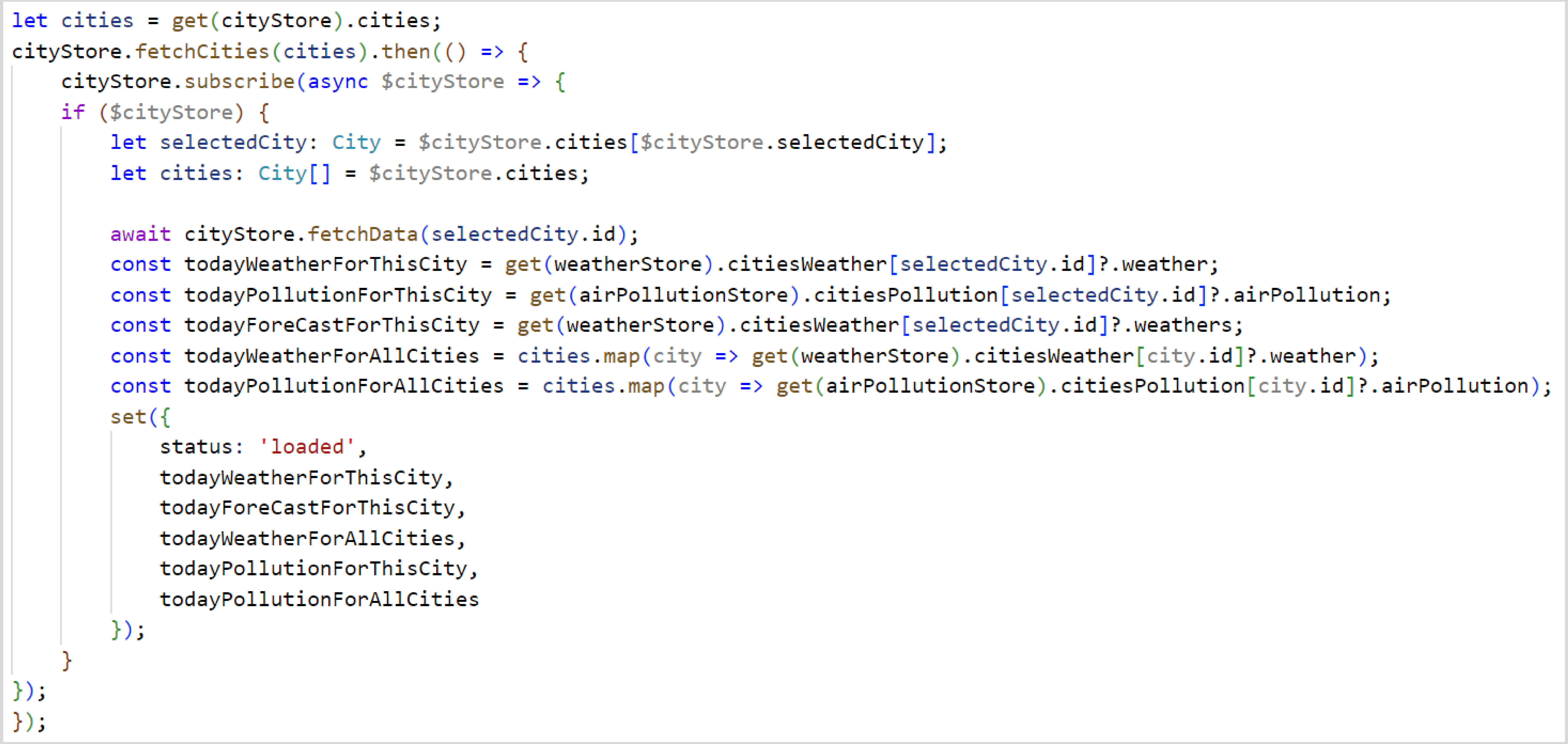
```
const { subscribe, set, update } = writable<CityState>({
  cities: initialCities,
  selectedCity: 0,
});

 const fetchData = async (id) => {
   await Promise.all([
           weatherStore.fetchWeather(id),
           weatherStore.fetchWeathers(id),
           airPollutionStore.fetchAirPollution(id)
         ])

}
```

Store: storeConnector.ts



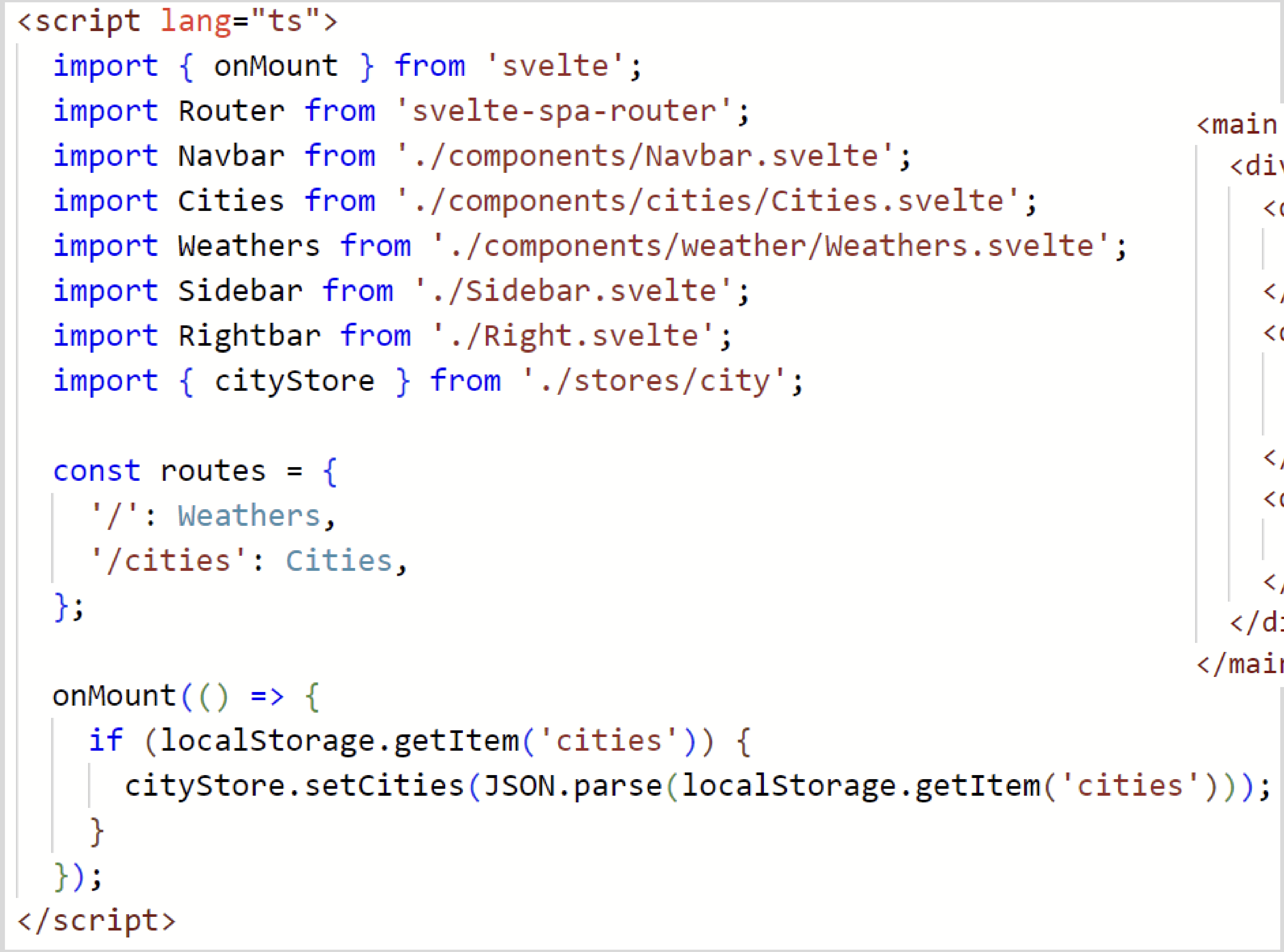
```
const createConnectorStore = () => {
    const { subscribe, update, set } = writable({
        status: 'loading',
        todayWeatherForThisCity: undefined,
        todayForeCastForThisCity: undefined,
        todayWeatherForAllCities: undefined,
        todayPollutionForThisCity: undefined,
        todayPollutionForAllCities: undefined
    });

     const setLoading = () => {
        update(state => ({ ...state, status: 'loading' }));
    };
```

Store: storeConnector.ts

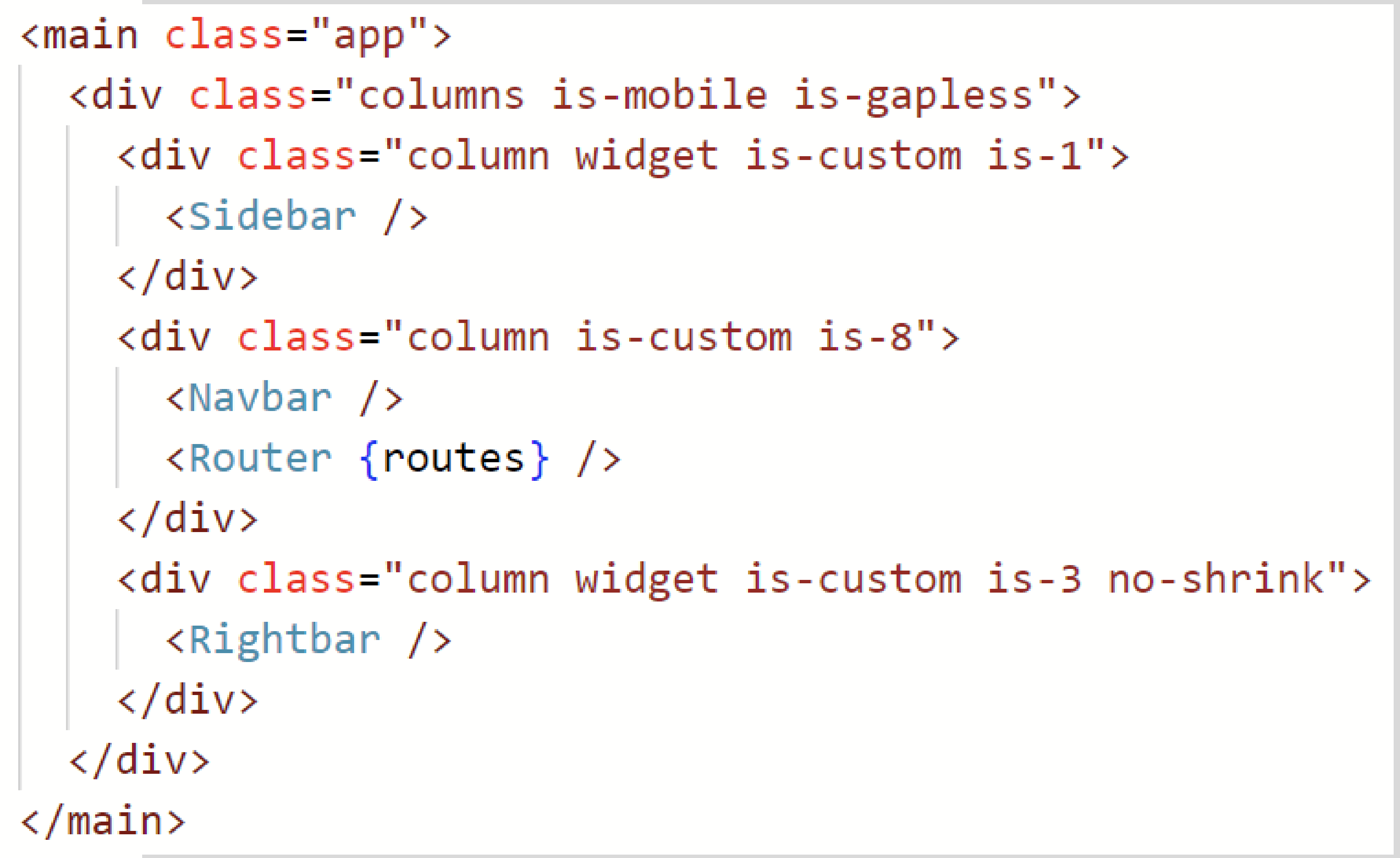


```
let cities = get(cityStore).cities;
cityStore.fetchCities(cities).then(() => {
    cityStore.subscribe(async $cityStore => {
    if ($cityStore) {
        let selectedCity: City = $cityStore.cities[$cityStore.selectedCity];
        let cities: City[] = $cityStore.cities;

        await cityStore.fetchData(selectedCity.id);
        const todayWeatherForThisCity = get(weatherStore).citiesWeather[selectedCity.id]?.weather;
        const todayPollutionForThisCity = get(airPollutionStore).citiesPollution[selectedCity.id]?.airPollution;
        const todayForeCastForThisCity = get(weatherStore).citiesWeather[selectedCity.id]?.weathers;
        const todayWeatherForAllCities = cities.map(city => get(weatherStore).citiesWeather[city.id]?.weather);
        const todayPollutionForAllCities = cities.map(city => get(airPollutionStore).citiesPollution[city.id]?.airPollution);
        set({
            status: 'loaded',
            todayWeatherForThisCity,
            todayForeCastForThisCity,
            todayWeatherForAllCities,
            todayPollutionForThisCity,
            todayPollutionForAllCities
        });
    }
});
});
```

Componentes: App.svelte y Router

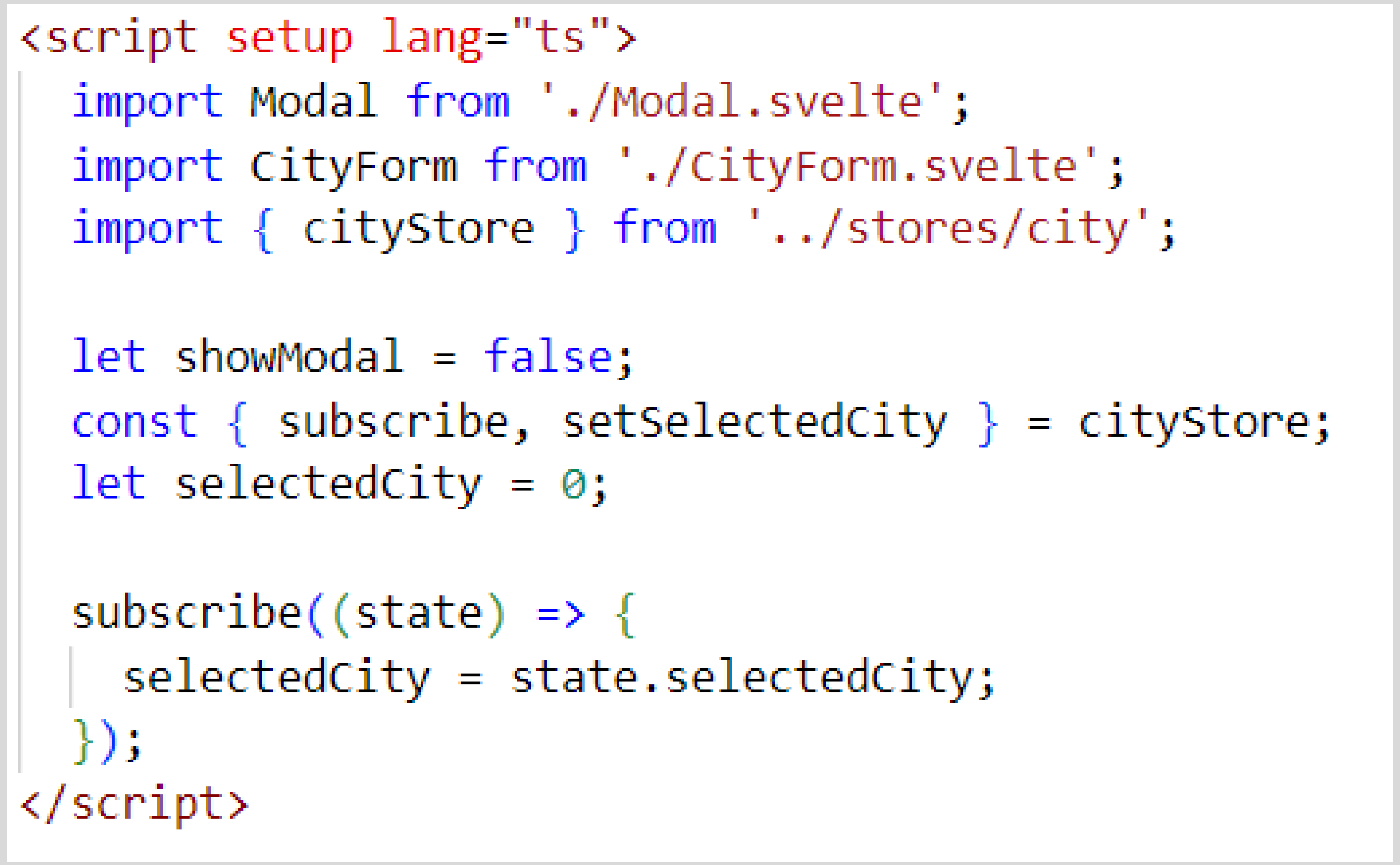


```
<script lang="ts">
  import { onMount } from 'svelte';
  import Router from 'svelte-spa-router';
  import Navbar from './components/Navbar.svelte';
  import Cities from './components/cities/Cities.svelte';
  import Weathers from './components/weather/Weathers.svelte';
  import Sidebar from './Sidebar.svelte';
  import Rightbar from './Right.svelte';
  import { cityStore } from './stores/city';

  const routes = {
    '/': Weathers,
    '/cities': Cities,
  };

  onMount(() => {
    if (localStorage.getItem('cities')) {
      cityStore.setCities(JSON.parse(localStorage.getItem('cities')));
    }
  });
</script>
```

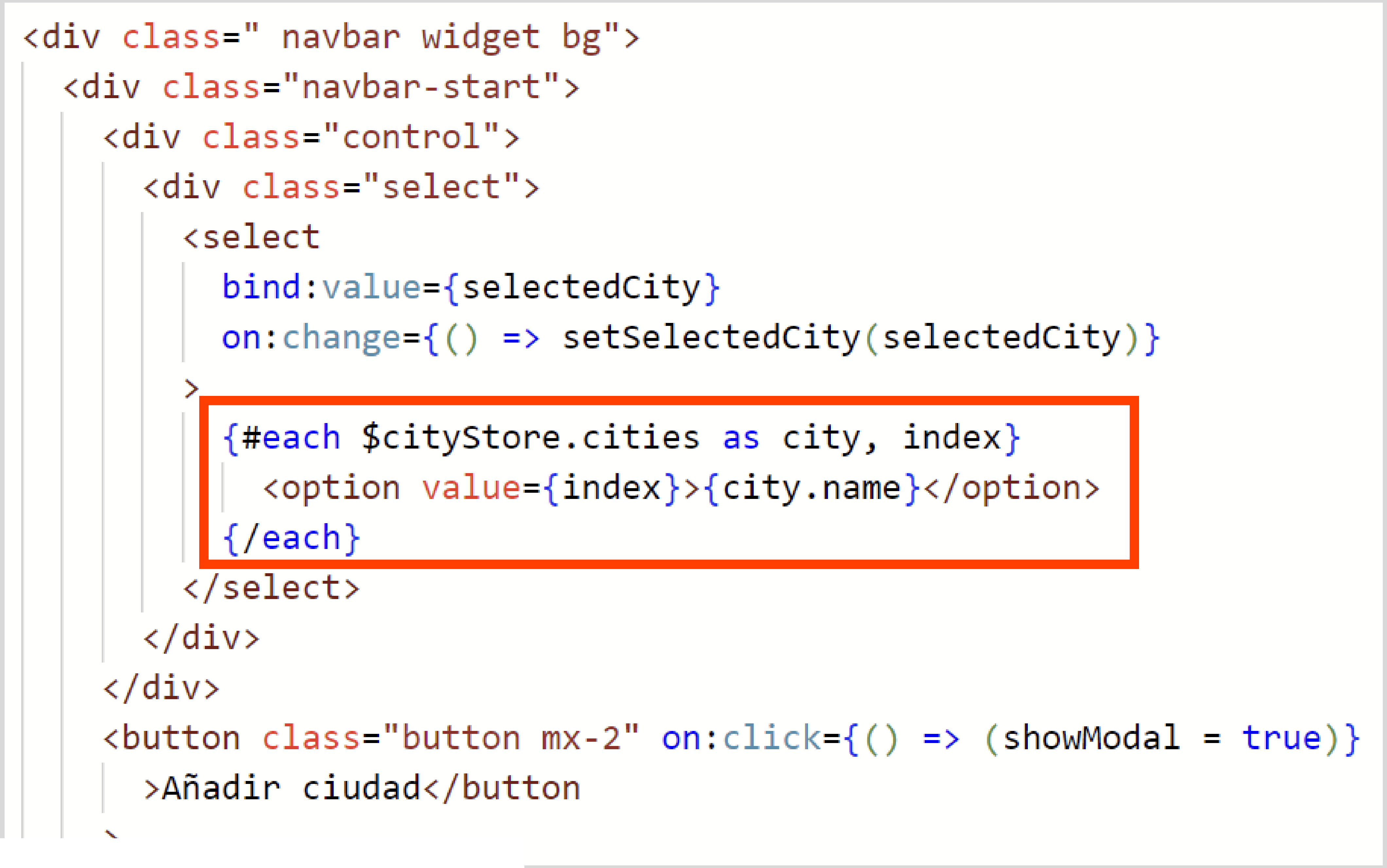


```
<main class="app">
  <div class="columns is-mobile is-gapless">
    <div class="column widget is-custom is-1">
      <Sidebar />
    </div>
    <div class="column is-custom is-8">
      <Navbar />
      <Router {routes} />
    </div>
    <div class="column widget is-custom is-3 no-shrink">
      <Rightbar />
    </div>
  </div>
</main>
```

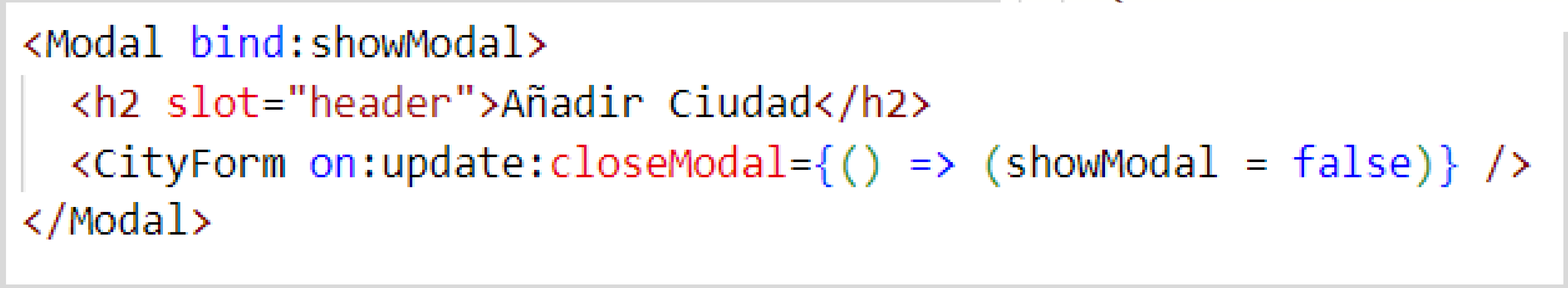
Componentes: Navbar.svelte



```
<script setup lang="ts">
  import Modal from './Modal.svelte';
  import CityForm from './CityForm.svelte';
  import { cityStore } from '../stores/city';

  let showModal = false;
  const { subscribe, setSelectedCity } = cityStore;
  let selectedCity = 0;

  subscribe((state) => {
    selectedCity = state.selectedCity;
  });
</script>
```

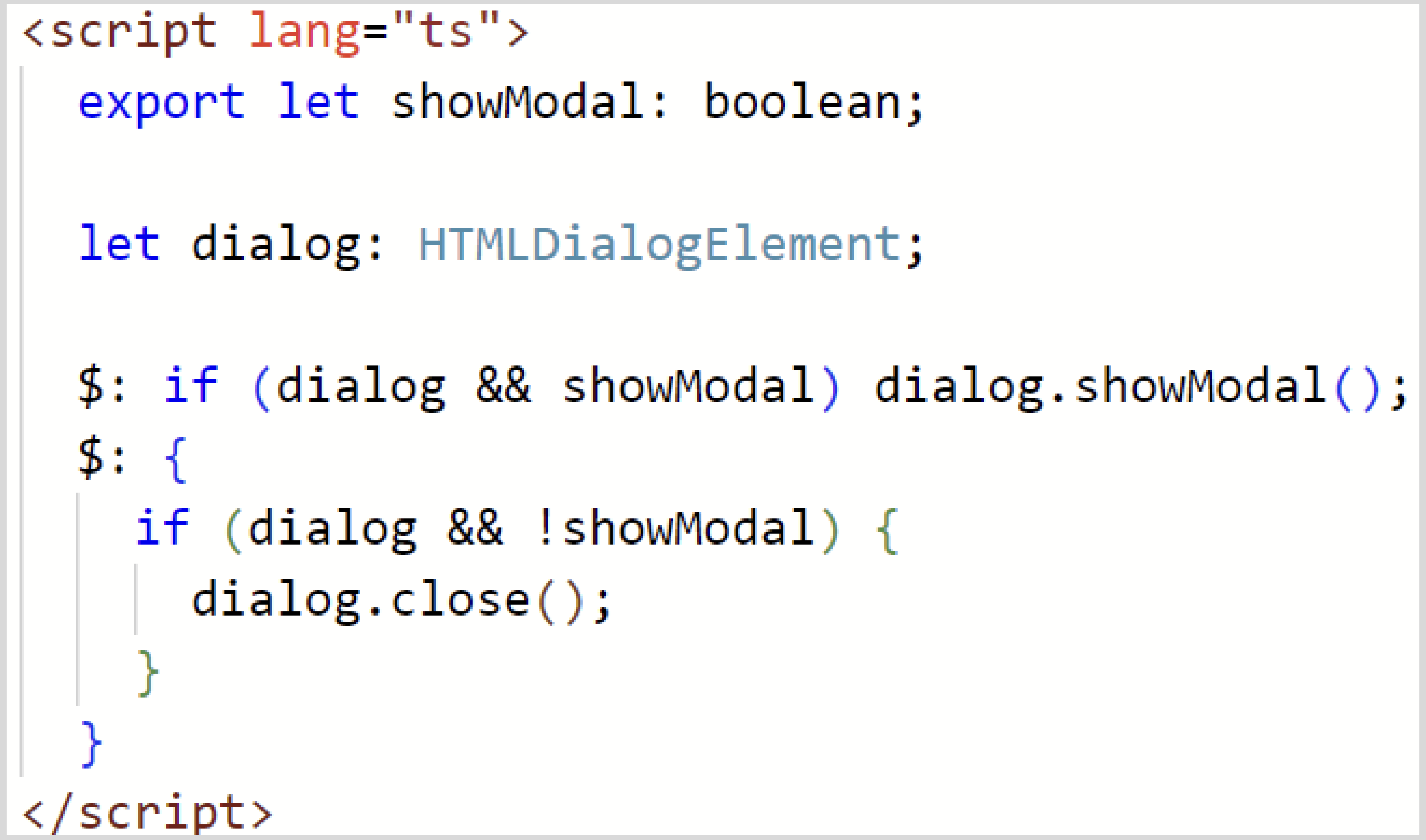


```
<div class=" navbar widget bg">
  <div class="navbar-start">
    <div class="control">
      <div class="select">
        <select
          bind:value={selectedCity}
          on:change={() => setSelectedCity(selectedCity)}
        >
          {#each $cityStore.cities as city, index}
            <option value={index}>{city.name}</option>
          {/each}
        </select>
      </div>
    </div>
    <button class="button mx-2" on:click={() => (showModal = true)}
      >Añadir ciudad</button
```

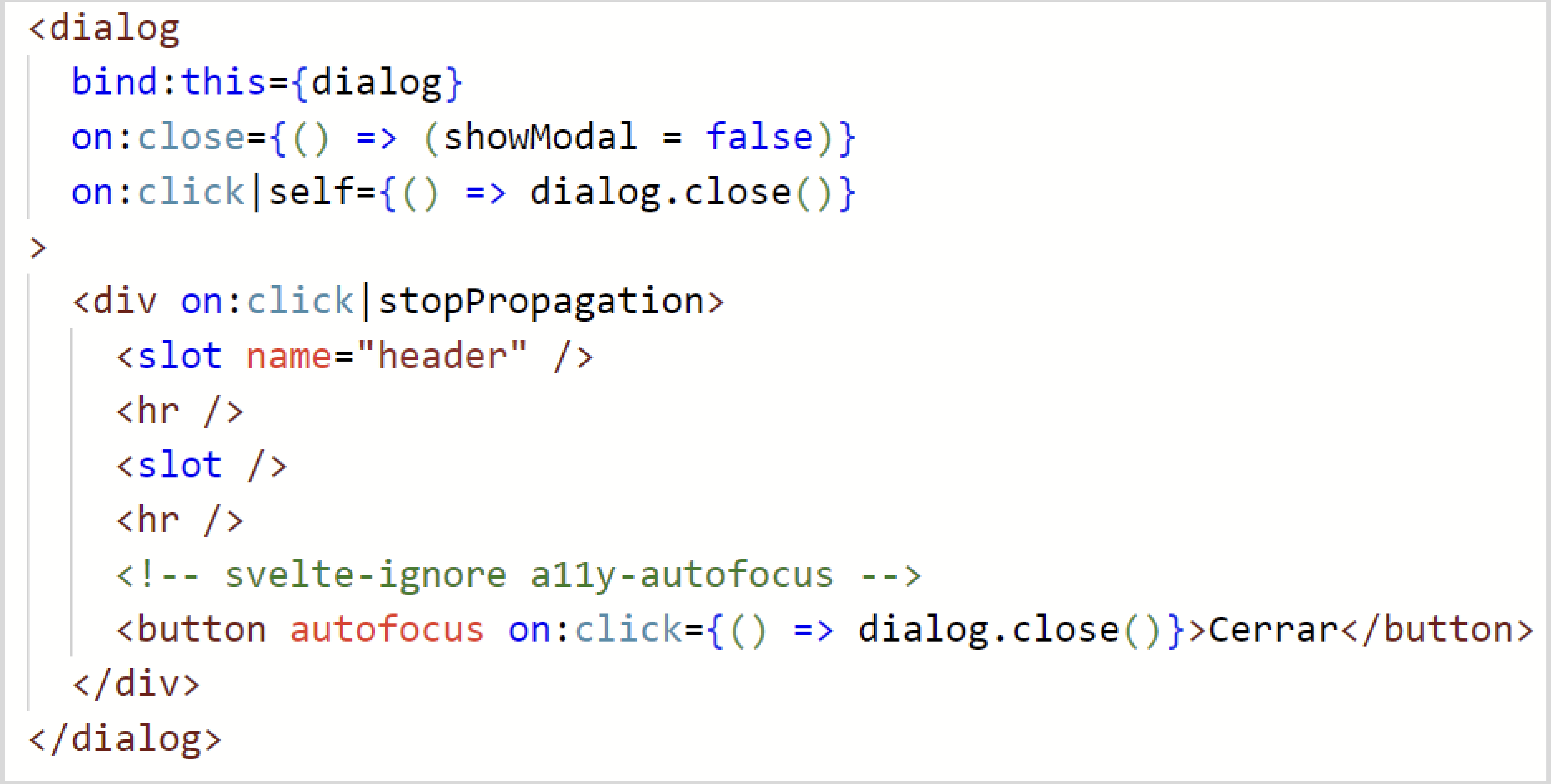


```
<Modal bind:showModal>
  <h2 slot="header">Añadir Ciudad</h2>
  <CityForm on:update:closeModal={() => (showModal = false)} />
</Modal>
```

Componentes: Modal.svelte



```svelte
<script lang="ts">
  export let showModal: boolean;

  let dialog: HTMLDialogElement;

  $: if (dialog && showModal) dialog.showModal();
  $: {
    if (dialog && !showModal) {
      dialog.close();
    }
  }
</script>
```



```svelte
<dialog
  bind:this={dialog}
  on:close={() => (showModal = false)}
  on:click|self={() => dialog.close()}
>
  <div on:click|stopPropagation>
    <slot name="header" />
    <hr />
    <slot />
    <hr />
    <!-- svelte-ignore a11y-autofocus -->
    <button autofocus on:click={() => dialog.close()}>Cerrar</button>
  </div>
</dialog>
```

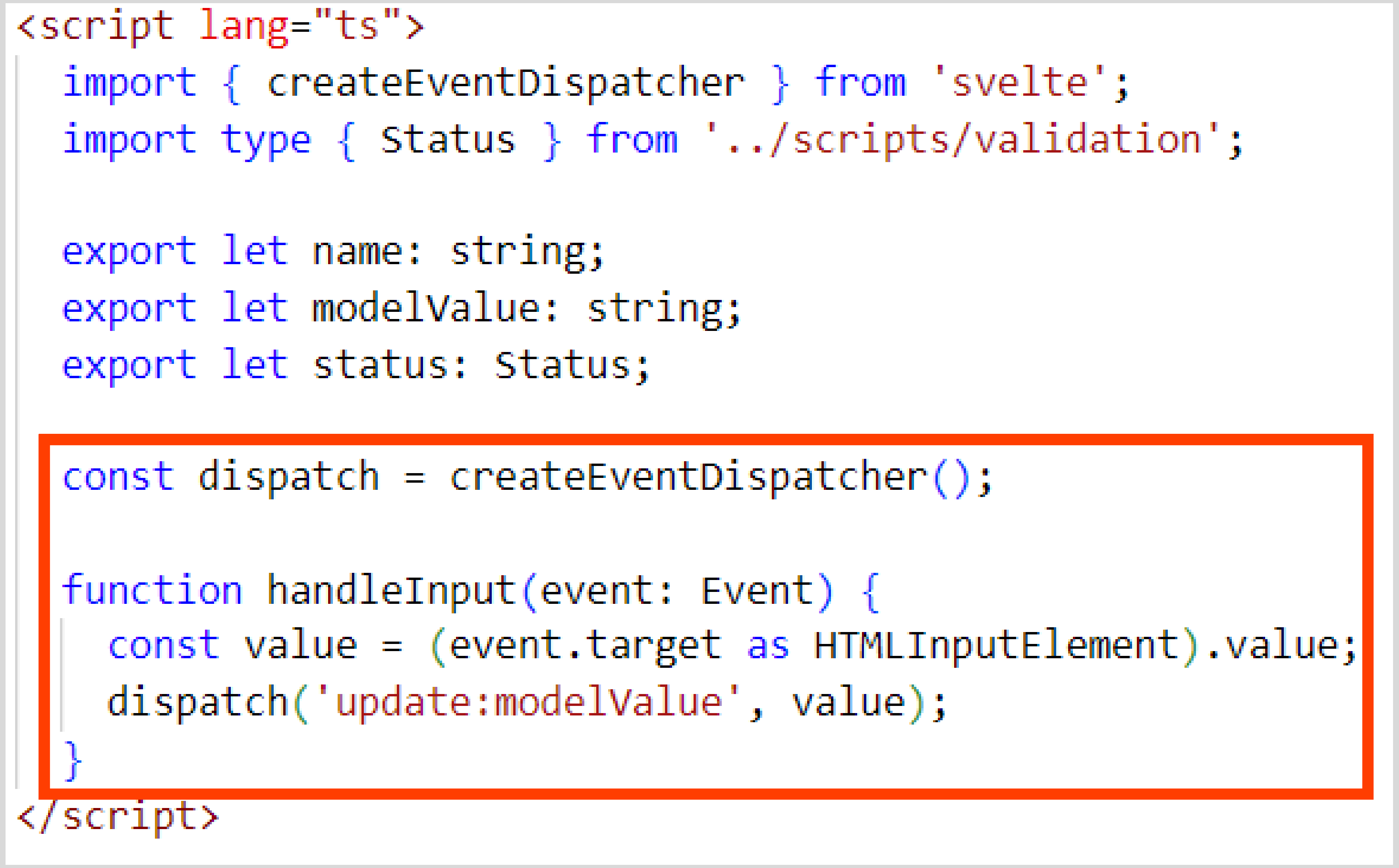
Componentes: CityForm.svelte

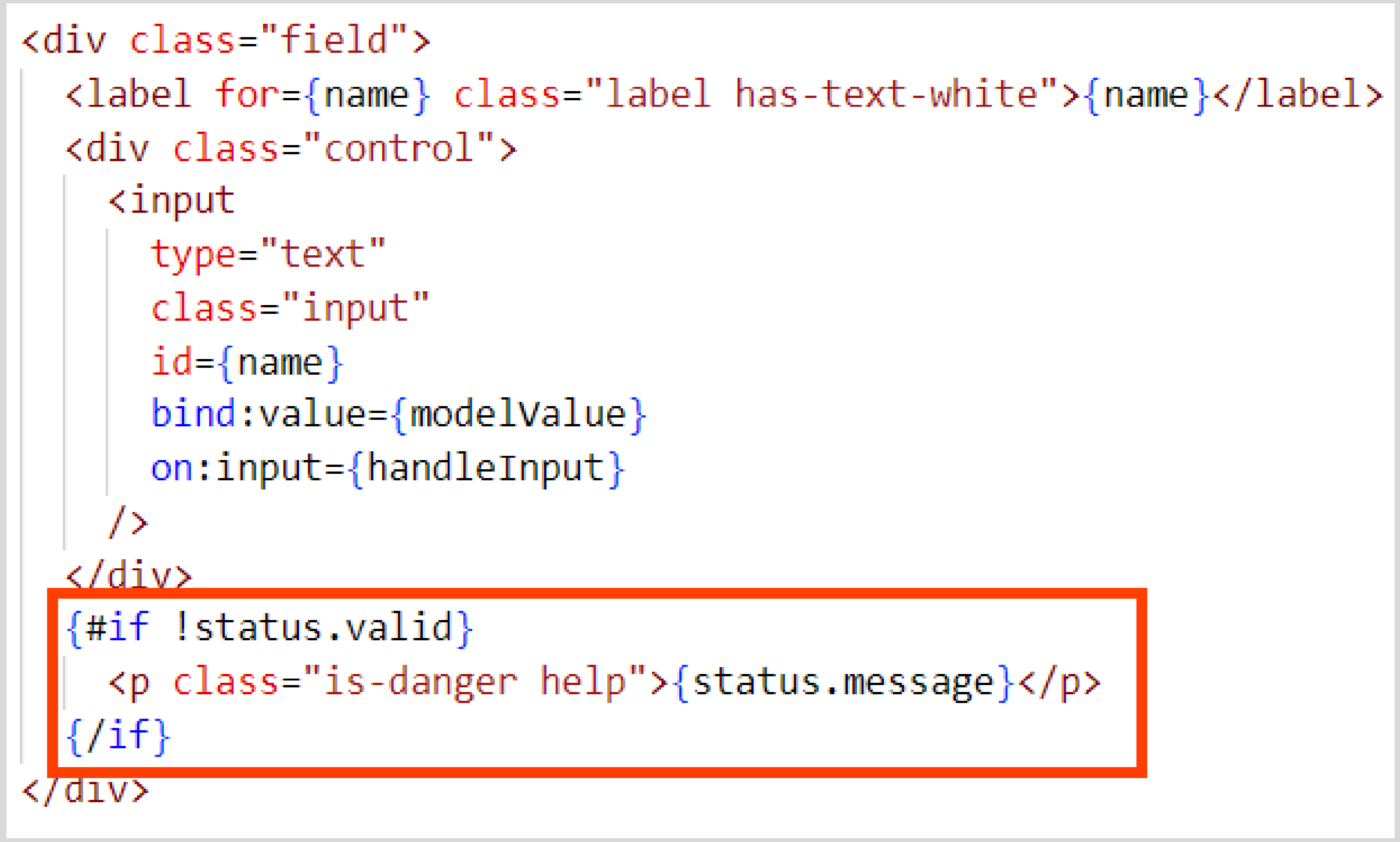
```
let nameValue: string;
let nameStatus: Status;
const name = writable('');

name.subscribe((value) => (nameValue = value));

name.subscribe((value) => {
  const { valid, message } = validate(value, [required]);
  nameStatus = { valid, message };
});
```

```
<form class="form" on:submit|preventDefault={handleSubmit}>
  <FormInput
    name="Nombre de la ciudad"
    bind:modelValue={nameValue}
    bind:status={nameStatus}
    on:update:modelValue={(event) => name.set(event.detail)}
  />
  <FormInput ···
  />
```

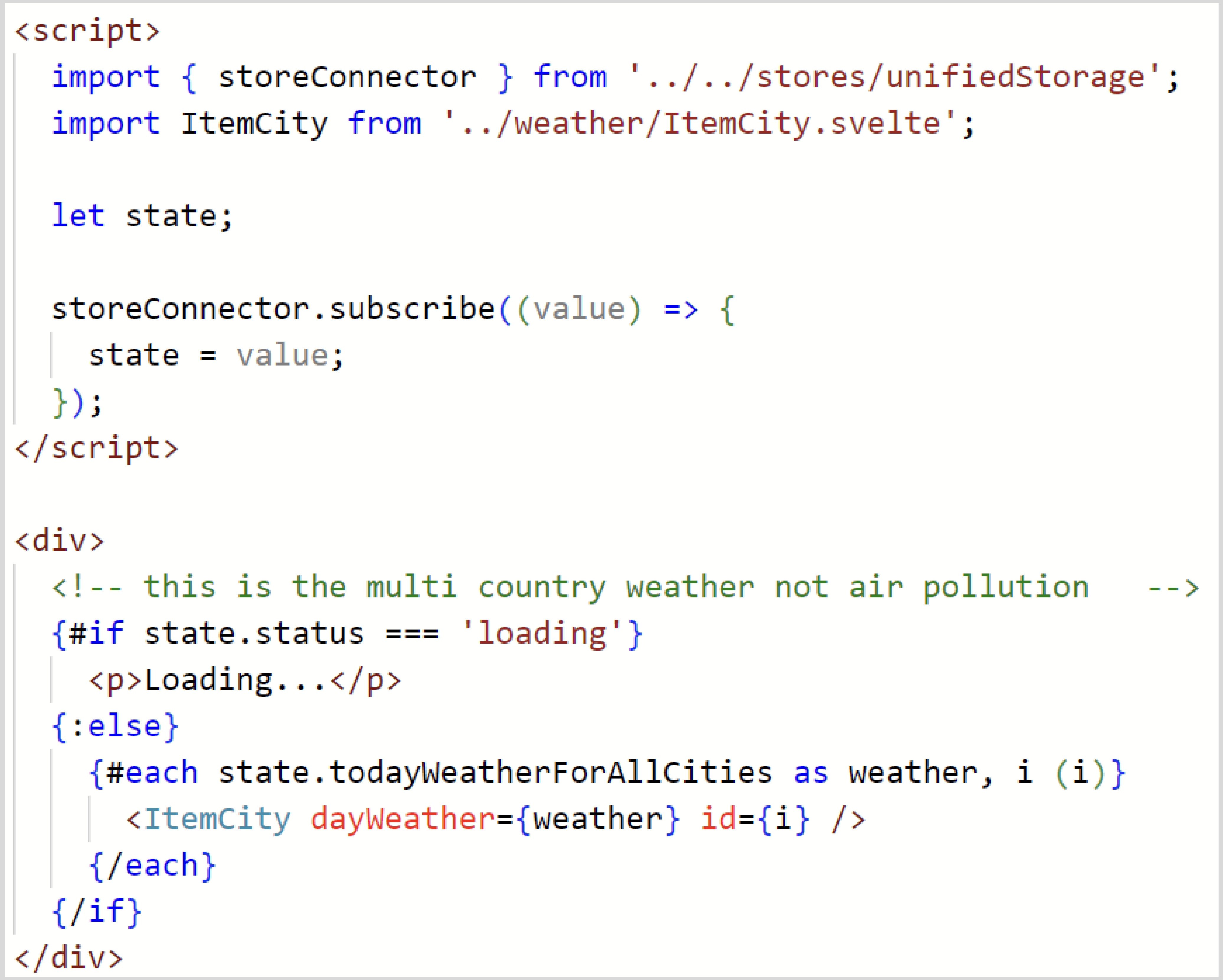
```
const dispatch = createEventDispatcher();

async function handleSubmit() {
  let cityLength: number;
  cityStore.subscribe((value) => {
    cityLength = value.cities.length;
  });

  const newCity: City = {
    name: nameValue,
    country: countryValue,
    latitude: Number(latitudeValue),
    longitude: Number(longitudeValue),
    imageURL: imageURLValue,
    id: cityLength,
  };
  storeConnector.setLoading();
  cityStore.addCity(newCity);

  dispatch('update:closeModal', false);
```

Componentes: FormInput.svelte


```
<script lang="ts">
  import { createEventDispatcher } from 'svelte';
  import type { Status } from '../scripts/validation';

  export let name: string;
  export let modelValue: string;
  export let status: Status;

  const dispatch = createEventDispatcher();

  function handleInput(event: Event) {
    const value = (event.target as HTMLInputElement).value;
    dispatch('update:modelValue', value);
  }
</script>
```


```
<div class="field">
  <label for={name} class="label has-text-white">{name}</label>
  <div class="control">
    <input
      type="text"
      class="input"
      id={name}
      bind:value={modelValue}
      on:input={handleInput}
    />
  </div>
  {#if !status.valid}
    <p class="is-danger help">{status.message}</p>
  {/if}
</div>
```

Componentes: Cities.svelte



```
<script>
  import { storeConnector } from '../../stores/unifiedStorage';
  import ItemCity from '../weather/ItemCity.svelte';

  let state;

  storeConnector.subscribe((value) => {
    state = value;
  });
</script>

<div>
  <!-- this is the multi country weather not air pollution   -->
  {#if state.status === 'loading'}
    <p>Loading...</p>
  {:else}
    {#each state.todayWeatherForAllCities as weather, i (i)}
      <ItemCity dayWeather={weather} id={i} />
    {/each}
  {/if}
</div>
```

CONCLUSIONES

TS

OpenWeather

Equipo 09